# Customizable Components

| Component | Description |
|---|---|
| Z_ACCESS_DATA | Read several records |
| Z_ACTIVATE_PREL_REC | Transfer preliminary data into database. Close the transaction |
| Z_ADD_PREL_REC | Add preliminary record |
| Z_ASSIGN_DEFAULT_KEY | Define preliminary key ID for a new object |
| Z_ASSIGN_INPUT_TO_KEY | Transfer the user input into internal variable |
| Z_ASSIGN_SUBDIALOG | Allocate dependent subdialogs to the higher-level dialog |
| Z_CHECK_EXISTENCE | Check existence of a record |
| Z_CLEAR_INPUT_FIELDS | Reset input fields |
| Z_CMD_EXEC_END | Any functionality after execution of a command |
| Z_CMD_EXEC_START | Any functionality before execution of a command |
| Z_CUSTOM_CMD | Execute any individual commands |
| Z_DELETE | Delete record |
| Z_FILL_DIALOG | Fill dialog elements |
| Z_FILL_ITEM | Transfer data base fields into list box |
| Z_INITIALIZE | Start processing |
| Z_LOCK_RECORD | Lock data record |
| Z_NAVIGATE_ON_ERROR | In case of error, set focus to field in error |
| Z_PASS_KEY | Pass key to internal variable |
| Z_PROCESS_ITEM | Process selected item |
| Z_READ_PREL_REC | Read preliminary record |
| Z_RECEIVE_DATA | Take data from modal window |
| Z_RECEIVE_KEY | Take foreign key from key dialog |
| Z_RETURN_KEY | Provide selected value to higher-level dialog |
| Z_RETURN_PARMS | Provide parameters (user buffer) to higher-level dialog |
| Z_SET_KEY_RANGE | Set key range for reading data |
| Z_SELECT | Transfer selected key ID into input fields |
| Z_UPDATE | Modify original data (if working without preliminary records) |
| Z_UPDATE_ITEM | Update the list box using data sent by another dialog |
| Z_UPDATE_PREL_KEY | Update the key ID of the preliminary copies after the ’Save as ...’ command |
| Z_UPDATE_PREL_REC | Update the preliminary copies |
| Z_VALIDATE | Validate user input |

## Z_ACCESS_DATA

Read multiple records.

The component usually contains the call to a multiple-object access module. After the call of the multiple-object access module, the subroutine Z_MR_ACCESS_RESULTS should be called. It sets frame variables based on the contents of the standard parameter data area of the access module. In addition, the standard error processing of a multiple-object access module is carried out by it in case of error.

For accesses which do not use the access modules created using the frame gallery, the variable LZ_REC_EOD must be set to FALSE if further data are present. The number of records found is to be assigned to the variable LZ_REC_NUM_FOUND.

## Z_ACTIVATE_PREL_REC

Transfer preliminary data into the data base.

The component usually contains the call to the frame subroutine Z_ACTIVATE_PREL_STD. From this subroutine, among other things, the activation module assigned to the variable LZ_ACTIVATE_MODULE in the component Z_INITIALIZE is called.

You can, if necessary, implement your own activation logic in this component instead of the call to the frame subroutine.

## Z_ADD_PREL_REC

Add preliminary record.

The frame has already preset the operation code and the key ID value for the storing of the preliminary data record. If you need values other than these, you can assign the appropriate values at the beginning of the component.

The suggested code of this component is tailored to the layout of a preliminary data record.

If you wish to add further preliminary data records, you can insert the following suggested code at the end of the component and adapt it accordingly:

```
INCLUDE ZXFXESCC      /* escape after error
RESET PZ_XAS000
      PZ_PRELIMINARY_VIEW
MOVE key       TO LZ_STD.LZ_PREL_KEY
MOVE #ZPL_SAVE TO PZ_AS_OPERATION
PERFORM new_ACCESS_PREL
```

## Z_ASSIGN_DEFAULT_KEY

Definition of the preliminary key ID when adding a new record.

This component defines the preliminary key ID of a newly added object before it is stored for the first time. It is possible to assign different values to the key ID displayed in the title line and to the technical key ID.

This component is only called if the variable LZ_USE_DEFAULT_KEY is set to TRUE.

## Z_ASSIGN_INPUT_TO_KEY

Transfer the user input into the variable LZ_SELECT_KEY.

This subroutine is only called if the variable LZ_START_USER_INPUT is set to TRUE.

## Z_ASSIGN_SUBDIALOG

Allocate Natural objects used.

In this component, the Natural object names of the subdialogs to be managed by the frame are made known to it. In addition, a local command is allocated in a table to each subdialog.

## Z_CHECK_EXISTENCE

Check existence of the record to be processed.

The frame has already preset the operation code for the existence checking. If you need another operation code, you can assign this at the beginning of the component.

The suggested code of this component is tailored to the checking of a data record.

If you wish to check further data records, you can insert the following suggested code at the end of the component and adapt it accordingly:

```
  INCLUDE ZXFXESCC      /* escape after error
  RESET PZ_XAS000
        PZ_MSG
  MOVE LZ_XA_READ TO PZ_AS_OPERATION
  MOVE key        TO P_view.xxx_ID

  CALLNAT ’xxxAS00N’ USING PZ_XAS000 P_view

  PERFORM Z_CHECK_RSP_CHECK_EXIST
```

## Z_CLEAR_INPUT_FIELDS

Reset input fields.

This component resets the values of all dialog elements and the ’linked variables’ connected to them. It is run through before the display of data newly read in and also after the command Z_CLEAR.

## Z_CMD_EXEC_END

User exit for any processing after execution of a command.

This component is run through after the execution of every command. If necessary, you can code here any processing that should be run through after a command.

The variable LZ_STD.LZ_CMD_ID contains the command originally invoked, while the variable LZ_STD.LZ_FRAME_CMD_ID contains the command to be processed by the frame. Using both these variables, you can control for which command which processing should be carried out.

## Z_CMD_EXEC_START

User exit for any processing before execution of a command.

This component is run through before the execution of every command. If necessary, you can code here any processing that should be run through before a command.

The variable LZ_STD.LZ_CMD_ID contains the command invoked. Using this variable, you can control for which command which processing should be carried out.

If you want to cause the frame to carry out another standard command instead of the command invoked, you can assign the required command to the variable LZ_STD.LZ_FRAME_CMD_ID in this component.

## Z_CUSTOM_CMD

User exit for commands that the frame does not support.

This component is always run through when a command is invoked that is not supported by the frame of the dialog affected. The variable LZ_STD.LZ_FRAME_CMD_ID contains the command invoked. Using this variable, you can control for which command which processing should be carried out.

## Z_DELETE

Delete record.

This is handled using either an access module or an activate module. If several data records are to be deleted, either several access modules or a special activate module is to be called. In case of error, error numbers are to be allocated to the array PZ_MSG_NUM and the value TRUE is to be allocated to LZ_VAL_ERR. By leaving the subroutine by ESCAPE ROUTINE, the accompanying error message is output in a message window.

## Z_FILL_DIALOG

Transfer the values from the interface of the access module to the dialog elements.

In this component, the variable values in the interface of the access module are transferred into the corresponding dialog elements and the ’linked variables’ connected to them. For simple ’input fields’, this can be done by a MOVE BY NAME statement. For complicated dialog elements, such as, control boxes or option buttons, it can be necessary to code this for each field individually.

## Z_FILL_ITEM

Transfer database fields into list box.

Since, in the usual case, more than one data record is read with Z_ACCESS_DATA, the record with the index LZ_REC_NUM_IND must be transferred into the frame variables LZ_BOX_ITEM and LZ_BOX_ITEM_KEY.

This is most easily handled using a MOVE BY NAME statement. For this, the two variables must be redefined to include the variables in the parameter data area of the access module.

To transfer the fields into the list box, the subroutine Z_ADD_ITEM must be called. In this way, it is possible in this subroutine to exclude data records from being transferred into the list box.

## Z_INITIALIZE

Initialize processing.

This component initializes variables used. Here you can undertake any individual initialization beyond the suggested code.

## Z_LOCK_RECORD

Lock records.

In this component, you lock the data records to be processed. The suggested code is so laid out that you lock the data record defined in the variable PZ_LOCAL.PZ_KEY. If you want to lock another data record, you can adapt the instructions appropriately.

If you want to lock a range of records, you must supply the variable LZ_LOCK_KEY with the beginning and the variable LZ_LOCK_KEY_END with the end of the range of keys.

If you want to lock several individual records or ranges of records, you can copy the suggested instructions and adapt them accordingly. With every call of the frame subroutine Z_CHECK_AND_LOCK_RECORD, a record or a range of records is locked.

## Z_NAVIGATE_ON_ERROR

Set focus to field in error or navigate to relevant dialog after an error is detected by the access layer. Head for the erroneous dialog element or dialog after the appearance of an error in the access layer.

This component contains a DECIDE statement in which, depending on parameters passed from the access layer. The focus is set to a field in error in the current dialog or the relevant subordinate dialog receives the focus.

If you use several object views in a dialog, you should embed the DECIDE statement of the suggested code in a decide statement which checks the object type:

```
DECIDE ON FIRST VALUE OF PZ_LOCAL.PZ_ERR_OBJ_ID
  VALUE objecttype 1
    DECIDE ON FIRST VALUE OF PZ_LOCAL.PZ_ERR_FLD_POS
      VALUE ....
          :
    END-DECIDE
  VALUE objecttype 2
    DECIDE ON FIRST VALUE OF PZ_LOCAL.PZ_ERR_FLD_POS
      VALUE ....
          :
    END-DECIDE
          :
END-DECIDE
```

## Z_PASS_KEY

Pass key to internal variable.

## Z_PROCESS_ITEM

Carry out processing for selected list box entry. As a rule, this is achieved by calling the subroutine Z_START_FUNCTION. Alternatively, you can code any special processing.

## Z_READ_PREL_REC

Read preliminary data record.

The frame has already preset the operation code and the key ID value to read the preliminary data record. If you need values other than these, you can assign the appropriate values at the beginning of the component.

The suggested code of this component reads a preliminary data record.

If you want to read further preliminary data records, you can insert the following suggested code at the end of the component and adapt it accordingly:

```
   INCLUDE ZXFXESCC      /* escape after error
   RESET PZ_XAS000
         PZ_PRELIMINARY_VIEW
   MOVE key        TO LZ_STD.LZ_PREL_KEY
   MOVE LZ_XA_READ TO PZ_AS_OPERATION
   PERFORM VIEW_ACCESS_PREL
```

## Z_RECEIVE_DATA

Accept data from modal window.

This component transfers data that were modified and confirmed in a modal window.

If, in a dialog, you receive different data from various modal windows, you must, when calling a modal window, note in a user-defined identifier which window was opened. When transferring the data into the component Z_RECEIVE_DATA, this identifier must be tested.

## Z_RECEIVE_KEY

Accept foreign key ID from key dialog.

This component accepts foreign key IDs that were selected in a key dialog.

If, in a dialog, you call selection helps for foreign key IDs of different fields, you must, when calling a selection help, note in a user-defined identifier for which field the selection help was opened. When accepting the key ID value, depending on this identifier, the appropriate field is to be used.

## Z_RETURN_KEY

Check user input for OK in the key dialog.

Validate input fields and transfer them into frame variable LZ_SELECT_KEY.

## Z_RETURN_PARMS

Provide parameters (user buffer) to a higher level dialog.

In this component, the data modified in a modal window can be transferred into the user buffer PZ_DATA. Subsequently, the modal window passes the user buffer to the higher level dialog. There, the data can be received in the component Z_RECEIVE_DATA.

## Z_SET_KEY_RANGE

Check start-value input and build up start value for multiple-record access module.

Example:

```
IF #IF_PLZ.STRING =  ''                  /* PLZ must be filled
   MOVE TRUE    TO LZ_VAL_ERR            /* error
   MOVE 4711    TO PZ_MSG.PZ_MSG_NUM(1)  /* message number
   MOVE #IF_PLZ TO LZ_FOCUS              /* position on input field
   ESCAPE ROUTINE                        /* no further processing
END-IF
*
*  build up start and end value for access module
*
MOVE #IF_PLZ.STRING   TO P_PERSONNEL.P_PERS_PLZ_FROM
MOVE #IF_PLZ.STRING   TO P_PERSONNEL.P_PERS_PLZ_THRU
MOVE #IF_NAME.STRING  TO P_PERSONNEL.P_PERS_NAME_FROM
```

## Z_SELECT

Transfer selected key ID into input field.

This component transfers the key ID values of the selected list box entry into the appropriate input fields.

## Z_UPDATE

Modify original data.

This component updates the data base according to the user input in dialogs that work without preliminary records.

## Z_UPDATE_ITEM

Update the list box based on a data record sent from another dialog.

The data record is to be transferred from the array PZ_LOCAL.PZ_DATA of the standard PDA ZXXLOC0A into the appropriate region of the parameter data area of the single-record access module. Afterwards, the individual fields are transferred into the corresponding fields of LZ_BOX.

## Z_UPDATE_PREL_KEY

Update the key ID of the preliminary files after the ’Save as ...’ command.

In this component, the preliminary files are read and stored with the new key ID and new processing time stamp PTS. This is always necessary when the command ’Save as ...’ is executed.

The suggested code of this component processes a preliminary data record.

If, in your function, you process several preliminary data records, you can copy the suggested code of the component for each further preliminary data record. Additionally, the following suggested code is to be inserted between the copied suggested codes and adapted accordingly.

```
   INCLUDE ZXFXESCC      /* escape after error
   RESET PZ_XAS000
         PZ_PRELIMINARY_VIEW

   MOVE LZ_DLG.LZ_PTS_OLD TO PZ_LOCAL.PZ_PTS
   MOVE LZ_DLG.LZ_KEY_OLD TO LZ_STD.LZ_PREL_KEY
   MOVE #ZPL_CLOSE        TO PZ_AS_OPERATION
```

## Z_UPDATE_PREL_REC

Update the preliminary files.

In this component, values of the dialog elements and the ’linked variables’ connected to them are transferred into the variables of the interface of the access layer. For simple ’input fields’, this can be done by a MOVE BY NAME statement. For complicated dialog elements, such as, control boxes or option buttons, it can be necessary to code this individually.

The frame has already preset the operation code and the key ID value for reading the preliminary data record. If you need values other than these, you can assign the appropriate values at the beginning of the component.

The suggested code of this component updates a preliminary data record.

If, in your function, you process several preliminary data records, you can copy the suggested code of the component for each further preliminary data record. Additionally, the following suggested code is to be inserted between the copied suggested codes and adapted accordingly.

```
   INCLUDE ZXFXESCC      /* escape after error
   RESET PZ_XAS000
         PZ_PRELIMINARY_VIEW
   MOVE key        TO LZ_STD.LZ_PREL_KEY
   MOVE LZ_XA_READ TO PZ_AS_OPERATION
```

# Z_VALIDATE

Validate user input.

In this component, the user inputs of the current dialog are validated. If necessary, data records referenced through foreign key IDs can also be read here to display additional information to this.